The Busy Platform Engineers Guide to API Gateways

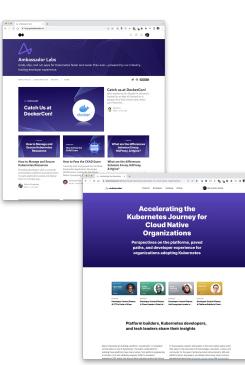
Daniel Bryant | Head of DevRel, Ambassador Labs Independent Consultant @danielbryantuk



- Choosing (or migrating) an API gateway is a Type 1 decision
- Traffic management is a key part of any platform
- Treat an API Gateway as a product
- Think about developer/operator experience
- Focus on workflows and tooling interoperability

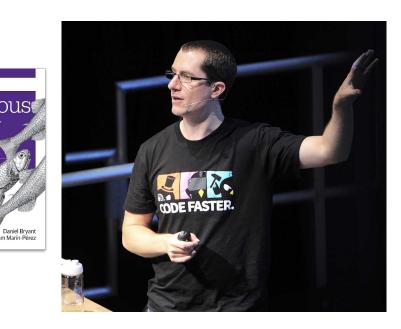


@danielbryantuk (he/him)



ambassador





Ambassador Edge Stack

linktr.ee/danielbryantuk

@danielbryantuk | @ambassadorlabs

Decisions, decisions, decisions...🤔



Software engineering is all about decisions

"Some decisions are **consequential and irreversible or nearly irreversible** – one-way doors – and these decisions must be made methodically, carefully, slowly, with great deliberation and consultation.

If you walk through and don't like what you see on the other side, you can't get back to where you were before. **We can call these Type 1 decisions**."

-Jeff Bezos, Founder of Amazon

Software engineering is all about decisions

"But most decisions aren't like that – they are **changeable**, **reversible** – they're two-way doors. If you've made a suboptimal **Type 2 decision**, you don't have to live with the consequences for that long. You can reopen the door and go back through.

Type 2 decisions can and should be made quickly by high judgment individuals or small groups."

-Jeff Bezos, Founder of Amazon

Choosing an API gateway is a type 1 decision

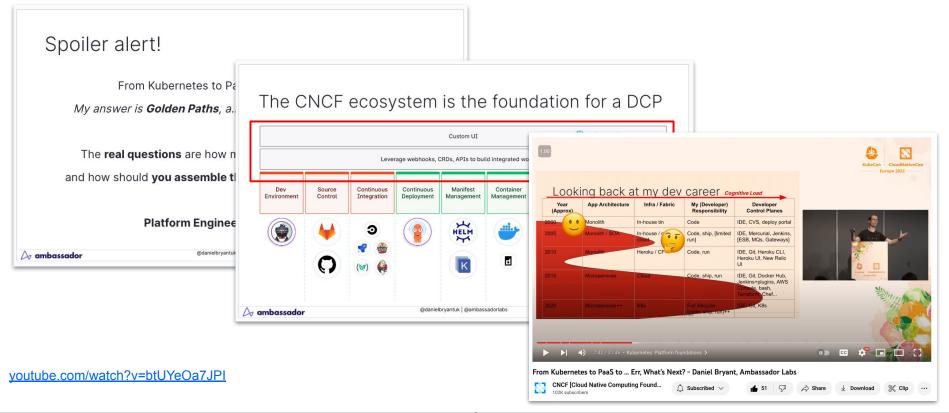
- An API gateway is difficult to change/replace
 - "Sticky" technology with a steep learning curve
- On the (business critical) hot path for all traffic
 - All user requests flow through this component
- Can be expensive
 - Contract lock-in is real, yo!
 - Platform engineers need to consider €€€ (especially now)

Previously at KubeCon '22





From Kubernetes to PaaS to... err, what's next?



A ambassador

@danielbryantuk | @ambassadorlabs

A quick recap for building platforms

From Kubernetes to PaaS to... err, what's next? My answer is **Golden Paths**, a.k.a. paved roads/paths/platforms

The **real questions** are how much should **you build yourself**, and how should **you assemble the control plane** for effective use?

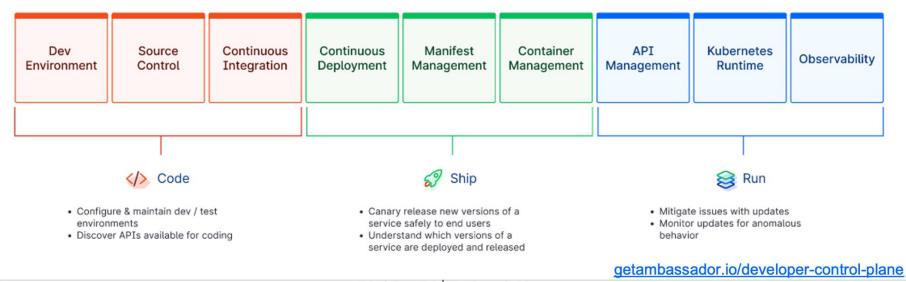
Platform Engineering is a how you do this



Looking back at my dev career cognitive Load

Year (Appr	App Architecture	Infra / Fabric	My (Developer) Responsibility	Developer Control Planes			
2000	Monolith	In-house tin	Code	IDE, CVS, deploy portal			
2005	Monolith / SOA	In-house /	Code, ship, [limited run]	IDE, Mercurial, Jenkins, [ESB, MQs, Gateways]			
2010	Monolith	Heroku / CF	Code, run	IDE, Git, Heroku CLI, Heroku UI, New Relic UI			
2015	Microservices	Cloud	Code, ship, run	IDE, Git, Docker Hub, Jenkins+plugins, AWS Console, bash, Terraform, Chef.			
2020	Microservices++	K8s	Full lifecycle (code, ship, run)++	IDE, Git, K8s			

The need for a platform control plane emerges



A ambassador

@danielbryantuk | @ambassadorlabs

The CNCF ecosystem is the foundation

				Sackstage						
		Leve	rage webhooks, (CRDs, APIs to bui	ld integrated wo	rkflow	YAMI			
Dev Environment	Source Control	Continuous Integration	Continuous Deployment	Manifest Management	Container Managemen	API Management	Kubernetes Runtime	Observability		
		3		HELM						
	0			K						
				brvantuk @ambass		getambas		loper-control-p		

🛆 ambassador

@danielbryantuk | @ambassadorlabs

A word of caution with platforms



1 You Retweeted

Daniel Terhorst-North @tastapod@mastodon.social @tastapod

I'll say it again. #k8s is just J2EE for the cloud generation: overengineered, overly complex for 99% of use cases, but ubiquitous and mandated as a solution before we have even looked at the problem.

People selling complexity-as-a-service will always make money.

Daniel Bryant @danielbryantuk · Jun 22 It's only 10am and I've already bumped into three Kubernetes-based dev platforms via email and Twitter!

The good news is that they are all pitching a high-level value prop and (to a large degree) a complete/end-to-end solution.

This is an interesting battle space for 2023 🧲

11:22 AM · Jun 22, 2023 · 59.1K Views

twitter.com/tastapod/status/1671810856273707008

...

A ambassador

Building platforms: What did I learn?

Treat platform as a product



You can't have good DevX without good UX

Focus on workflows and tooling interoperability





Treat Platform API Gateway as a Product





API Gateway as a Product

- Beware of "product" vs "project"
 - API gateways need a long-term (product) owner
 - Staff and resource an API gateway appropriately
 - If you want build an API gateway, it has to be a product (but don't do this!)
- Take care when lifting and shifting an API Gateway
 - Nearly always end up replatforming ("lift-tinker-and-shift")

API Gateway as a Product

- Know your users!
 - API gateways have multiple users (personas)
 - Identify them and their requirements
 - Top down vs bottom up
 - User research is invaluable



"Many platform engineering teams I see operate under the 'build it and they will come' philosophy.

...

No, they f*cking won't"

@mipsytipsy at #QConNY riffing on the need to work closely with your customer -- the developers -- to understand requirements



10:48 PM · Jun 15, 2023 · **17.5K** Views

twitter.com/danielbryantuk/status/1669446786354692097

API Gateway as a Product

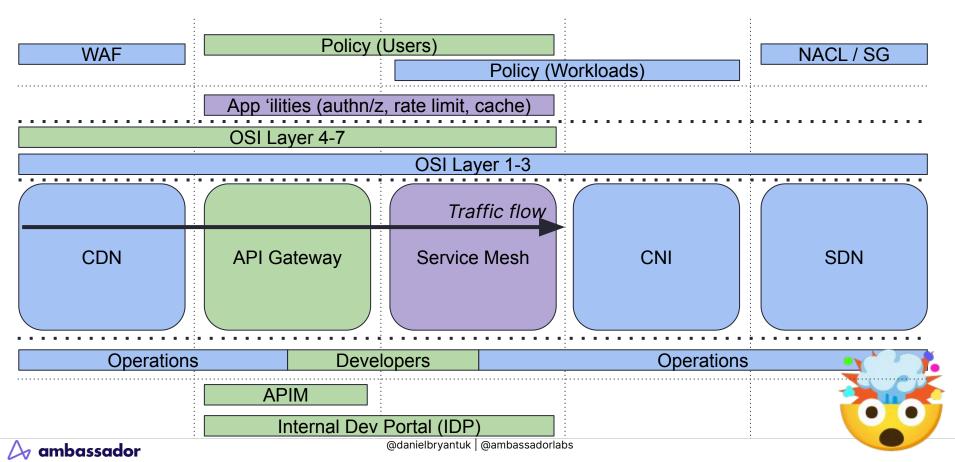
- Understand where the API gateway fits into the bigger solution
- The modern cloud native communication stack is complicated
 - The API gateway is only part of the solution





A ambassador

Modern cloud native comms stack



All-in-one or one-for-all?

- You can implement "all-in-one" solutions
 - o Solo
 - Isovalent
 - Kong
 - Cloud vendors(?)

• Or choose best of breed for each component

- Ambassador's Emissary-ingress, Envoy Gateway
- Buoyant's Linkerd, HashiCorp's Consul
- Cloud vendor CNI
- Cloudflare



You can't have good DevX without good UX

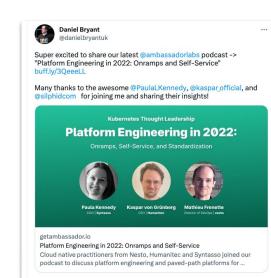


You can't have good DevX without good UX

- Understand the approach and defaults for your platform
 - Kubernetes native (CRDs, GitOps-friendly)
 - CLI or API-driven
 - UI-driven

ambassador

- Tailor the experience to personas
 - Developer experience
 - Operator experience
- Platform engineering tenet: self-service
 - But this means many things to many people
 - PRs vs biz-focused clickops



8:33 AM · Aug 10, 2022

Self-Service Configuration



```
apiVersion: getambassador.io/v3alpha1
kind: Mapping
metada
  apiVersion: getambassador.io/v3alpha1
s kind: Mapping
  meta
    na apiVersion: getambassador.io/v3alpha1
  spec kind: Mapping
    pr metadata:
         name: restricted-mapping
    se
    we spec:
         host: restricted.example.com
         prefix: /restricted/
         rewrite: /a/very/safe/path/
         rewrite_host: safe.example.com
         service: dangerous-service
```

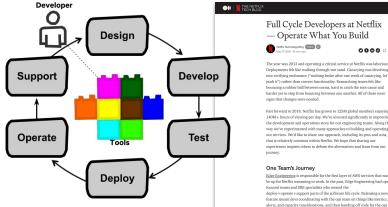
```
apiVersion: getambassador.io/v3alpha1
kind: Listener
metadat
  apiVersion: getambassador.io/v3alpha1
sp kind: Host
  met
      apiVersion: getambassador.io/v3alpha1
      kind: AuthService
      metadata:
        name: extauth-service
  SDe
      spec:
        auth_service: example-auth
        path prefix: "/extauth"
        allowed_request_headers:
          - "x-example-session"
        allowed_authorization_headers:
          - "x-example-session"
          - "x-example-userid"
```

Focus on workflows and tooling interoperability



Workflow and interop

"[The] centralized [platform] teams act as force multipliers by turning their specialized knowledge into reusable building blocks."



ambassador

Full Cycle Developers at Netflix - Operate What You Build Netflix Technology Blog 📧 🔒 0000 0 -The year was 2012 and operating a critical service at Netflix was laborious Deployments felt like walking through wet sand. Canarying was devolving into verifying endurance ("nothing broke after one week of canarying, let's push it") rather than correct functionality. Researching issues felt like bouncing a rubber ball between teams, hard to catch the root cause and harder yet to ston from houncing between one another. All of these were signs that changes were needed Fast forward to 2018. Netflix has grown to 125M global members enjoying 140M+ hours of viewing per day. We've invested significantly in improving the development and operations story for our engineering teams. Along the way we've experimented with many approaches to building and operating our services. We'd like to share one approach, including its pros and cons. that is relatively common within Netflix. We hope that sharing our experiences inspires others to debate the alternatives and learn from ou One Team's Journey Edge Engineering is responsible for the first layer of AWS services that must be up for Netflix streaming to work. In the past, Edge Engineering had ops focused teams and SRE specialists who owned the deploy+operate+support parts of the software life cycle. Releasing a new

> team to deploy and operate. To be effective at running the code and supporting partners, the ops teams needed ongoing training on new features and bug fixes. The primary upside of having a separate ons tean was less developer interrupts when things were going well.

netflixtechblog.com/full-cvcle-developers-at-netflix-a08c31f83249



"A good deal of the job is ultimately about finding the right balances between standardization and autonomy"

srvaroa.github.io/paas/infrastructure/platform/kubernetes/cloud/2020/01/02/talk-how-to-build-a-paas-for-1500-engineers.html

@danielbryantuk | @ambassadorlabs

Self-Service Configuration



```
apiVersion: getambassador.io/v3alpha1
kind: Mapping
metada
  apiVersion: getambassador.io/v3alpha1
s kind: Mapping
  meta
    na apiVersion: getambassador.io/v3alpha1
  spec kind: Mapping
    pr metadata:
         name: restricted-mapping
    se
    we spec:
         host: restricted.example.com
         prefix: /restricted/
         rewrite: /a/very/safe/path/
         rewrite_host: safe.example.com
         service: dangerous-service
```

```
apiVersion: getambassador.io/v3alpha1
kind: Listener
metadat
  apiVersion: getambassador.io/v3alpha1
sp kind: Host
  met
      apiVersion: getambassador.io/v3alpha1
      kind: AuthService
      metadata:
        name: extauth-service
  SDe
      spec:
        auth_service: example-auth
        path prefix: "/extauth"
        allowed_request_headers:
          - "x-example-session"
        allowed_authorization_headers:
          - "x-example-session"
          - "x-example-userid"
```

Separation of Concerns



```
apiVersion: getambassador.io/v3alpha1
kind: Mapping
metada
  apiVersion: getambassador.io/v3alpha1
s kind: Mapping
  metar
    na apiVersion: getambassador.io/v3alpha1
  spec kind: Mapping
    pr metadata:
         name: restricted-mapping
    se
    we spec:
         host: restricted.example.com
         prefix: /restricted/
         rewrite: /a/very/safe/path/
          ewrite_host: safe.example.com
          ervice: dangerous-service
```

ambassador

```
apiVersion: getambassador.io/v3alpha1
kind: Listener
metadat
  apiVersion: getambassador.io/v3alpha1
sd kind: Host
  met
      apiVersion: getambassador.io/v3alpha1
      kind: AuthService
      metadata:
        name: extauth-service
  SDe
      spec:
        auth_service: example-auth
        path prefix: "/extauth"
        allowed_request_headers:
            "x-example-session"
             ed_authorization_headers:
             x-example-session"
             x-example-userid"
```

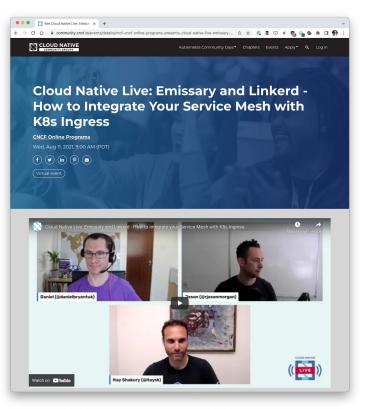
@danielbryantuk | @ambas

Extra validation when applying global configuration?



A ambassador

Interop Example: Emissary-ingress & Linkerd



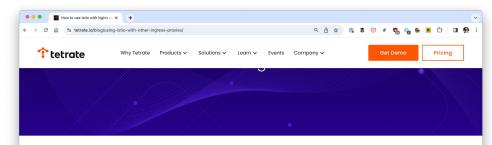
ambassador

CNCF projects

- Emissary-ingress: n/s gateway
- Linkerd: e/w service mesh
- Both use <u>Kubernetes Resource Model (KRM)</u>
 - CRDs, controllers, best practices
- One line integration

kubectl -n emissary get deploy emissary-ingress -o yaml | \
linkerd inject --skip-inbound-ports 80,443 - | \
kubectl apply -f -

• Similar configuration across projects

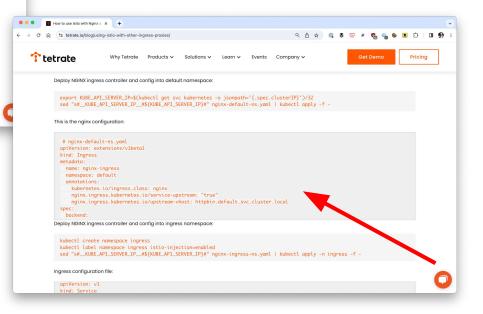


Here's How to Use Istio with Other Nginx Ingress Proxies

March 11, 2020 Authors: Tetrate



tetrate.io/blog/using-istio-with-other-ingress-proxies/

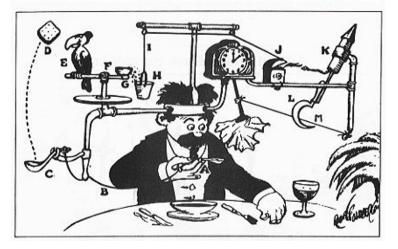


△ ambassador

API gateway plugins: love 'em/hate 'em

- Plugins/extension/filters provide:
 - Reusability
 - Separation of concerns
 - Performance (?)
- But, they are often highly coupled
 - With the API gateway
 - With the system itself
- Please, please, please don't put business logic in them!
 - I've seen this way too many times

Real-life API gateway plugin architecture



A ambassador





Tell me more about my (K8s) API Gateway options

C D & docs.google.com/sprea	idsheets/d/191Wi	VNpjJZza6-nbG4	ZoUMXMpUK8K	ClosvQ80f-oq3k/	edit/fgid=90773	1238										Ċ	й 🔃 🛢 🤅	/ 🗂 😘		1 69
Kubernetes Ingress Controllers															00	1000	0 0	E 0	Share	
File Edit View Insert Format Dat	a Tools Exten	sions Help																ч u.		
d ⊕ ¶ 100% + \$ % .0	.00 123 De	laul. • - [12 + B Z	÷ A Ò,	H	- + - p -	A * @ # #	- γ - Σ												
				-																
• ja																				
8 C	н	1	J	к	L	м	N	0	P	0	R	5	т	U	ll v	W	х	Y	z	
	Ingress		Apache	Azure App		HAProxy	HAProxy				Emissary-ing									
Product/Project	Nginx	Kong	APISIX	Gateway Ingress	Nginx+	Tech	(jcmoraisjr)	Voyager	Istio Ingress	Contour	(ex.Ambassadar)	Gloo	Traefik	Skipper	Citrix Ingress	GKE Ingress	ALB Ingress	AKO	KrakenD	
1. General info				ingreaa							(acressing)				(0)		
Based on	nginx	nginx	nginx	nginx + Azure	and the second sec	haproxy	haproxy	haproxy	envoy	envoy	envoy	erway	traefik	skipper		DC (NetScaler) In	- 0 / N	ISX Advanced	Lura Project	g
	-			App Gateway	nginx										github.co			LB (Avi)	Linux foundation	in)
Documentation 2. Protocols	16pg.//suberneses	https://docs.kong	e htps://gitub.con	https://azuro.githu	https://docs.ngin	< https://gittub.co	m https://haproxy-ing	https://www.agerm	reference	https://projectcon	E Ellps//www.gecan	https://docs.solo.	https://doc.traefik	https://opensouri	S D Citrix AD	(NetScaler) Ingress I	Controller for	gs://www.stworks	s https://www.krake	H DER
HTTPHTTPS	~	1	1	1	1	1	1	1	V	1	1	1	1	1	1	1	1	1	1	
HTTP2	1	1	1	Partial	1	V	J	1	1	V	V	1	1	×	1	1	1	1	1	
HTTP3 (QUIC)	Proview	*	*	*	Preview	×	×	×	×	×	v .	×	Preview	*	V	×	×	×	×	
gRPC	1	~	~	*	V	~	~	~	~	~	~	\checkmark	V	×	~	· · ·	~	V .	Partial	
TCP	Partial	~	~	*	~	1	1	1	~	~	~	~	1	×	~	~	~	V .	×	
TCP+TLS	×	~	· .	×	~	~	~	~	~	~	~	~	~	×	~	~	~	v .	×	
UDP	Partial	~	· .	*	~	×	×	×	×	×	×	×	~	×	~	~	v .	1	×	
Websockets	V	~	~	~	~	~	~	~	~	~	~	~	V	~	~	V .	v	V .	~	
Proxy Protocol	~	~	~	Needs help	~	~	~	~	~	~	· ~	~	~	×	×	×	~	~	×	
3. Clients																				
Rate limiting (L4)	V	1	Needs help	Partial	1	/		.1				.1	./		.1	1	Needs help	J .		
Rate limiting (L7)	1	1	, J	Partial	1							1					Needs help	1.		
WAF	1		Partial	-artial	V						ý .	V				· · ·	J	· .		
Timeouts	7	./	/		./	/		- 										, , ,		
Safe-list/Block-list	1	1			./			Partial			ý.	Needs help	3	/	./	· · ·		ý .		
Authentication	J.	1		*	J	1		~	· , ·	1	· · ·	~			1		ý.	, · ·	~	
Authorisation	*	1	/	×	J	J	Partial		1		. ·	1	J		1		1	1	1	
4. Traffic routing																			•	
Host	5	1	~	J	~	~	V	~	~	J	~	~	J	J	~	~	J	V	~	
Path	5	~		1	~	~		J	~	~	V .	~	~	~	~	~	~	~	~	
Headers	~	~	· ·	~	~	~	· ·	×	~	~	~	~	<i>v</i>	~	~	×	~	~	×	
Querystring	V	~	· _	~	~	~		×	~	×	✓ .	\checkmark	~	~	~	×	~	~	×	
Method	~	~	~	~	V	~		×	~	~	~	\checkmark	~	~	~	×	~	~	~	
ClientIP	1	~	~	~	~	~	· ·	×	~	×	\checkmark	\checkmark	~	~	~	×	~	~	×	
Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Notes	Note
5. Upstream probes/resiliency	Notes	Notes	Notes	NODES	NODES	Notes	Notes	Notes	Notes	NODES	Notes	Notes	Notes	Notes	NOORS	NODES	NOME	Notes	Notes	NOB
Healthchecks	×	1		V .	1	J		1	1	~	· .	1	1	×		V .	1	~	Partial	•
Retries	~	~		· · ·	1	1		*	1	1	· .	1	1	Partial		Needs help	×	ý ,	*	
Circuit Breaker	×	V		×	ý	. ,	. *	*	ý .	*	v .	1	7	J		Needs help	×	×	V	
6.Load balancer strategies					•															
Round robin	~	\checkmark	~	×	√	~	V	~	~	~	~	\checkmark	~	~	~	~	~	~	~	
Sticky sessions	7	~	1	×	V	V	V	~	~	1	~	~	~	Partial	~	~	~	~	×	
Least connections	×	\checkmark	~	×	~	~	~	~	~	~	~	\checkmark	×	×	~	\checkmark	~	~	×	
Ring hash	~	\checkmark	~	×	V	×	×	×	~	~	~	\checkmark	×	1	~	~	×	~	×	
Maglev	×	×	×	×	×	×	×	×	~	×	~	~	×	×	×	\checkmark	×	×	×	
Exponential-Weighted-Moving-Average	~	×	~	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	
Custom load balancing	×	×	~	×	Partial	×	×	×	×	×	×	×	×	×	×	×	×	×	×	
Power of two random choices	Needs help	Needs help	Needs help	Needs help	~	×	×	Needs help	Needs help	Needs help	Needs help	Needs help	Needs help	V	Needs help	Needs help	Needs help	Needs help	~	
Global load balancing	Needs help	Needs help	1	Needs help	1	×	Needs help	Needs help	Needs help	Needs help	Needs help	Needs help	Needs help	Needs help	1	Needs help	Needs help	1	×	

LearnK8s: https://docs.google.com/spreadsheets/d/191WWNpjJ2za6-nbG4ZoUMXMpUK8KIClosvQB0f-oq3k/edit?usp=sharing

A ambassador

Conclusion

- Choosing (or migrating) an API Gateway is a Type 1 decision
 - Tricky to reverse: but the right decision adds a lot of value
 - Clear ownership needed for platform and API gateway
- Treat API Gateways as a product
 - Identify personas and requirements
 - Integration within the wider cloud native comms stack is key
- Think about developer/operator experience
 - Self-service for the win!
- Focus on workflows and tooling interoperability
 - Good integration and appropriate extensions are the key!

Thank you!

@danielbryantuk

Improving Cloud Native DevEx: The API Gateway Perspective

Moving to the Cloud: Exploring the API Gateway to Success

Platform Engineering Guide

